**Listing 1. Funkcje odpowiedzialne za wysyłanie rozkazów lub danych obrazu**

void writeCommand**(**uint8\_t Command**)**

**{**

RESET\_DC**;** //DC=0 wskazuje na rozkaz sterujący

RESET\_WR**;**

DATA\_PORT **=** Command**;**

SET\_WR**;** //Zapis danych następuje na rosnącym zboczu sygnału WR

SET\_DC**;** //DC domyślnie ustawiine, by przyspieszyć transfer danych obrazu

**}**

void writeData**(**uint8\_t Data**)**

**{**

RESET\_WR**;**

DATA\_PORT **=** Data**;**

SET\_WR**;** //Zapis danych następuje na rosnącym zboczu sygnału WR

**}**

**Listing 2. Funkcja inicjalizacyjna sterownika SSD1327 wyświetlacza WEO128128**

void initOLED**(**void**)**

**{**

//Port danych jako wyjściowy, równy 0x00

DATA\_DDR **=** 0xFF**;**

//Port sterujący jako wyjściowy, równy 0xFF

CTRL\_PORT **|=** **(**1**<<**WR\_PIN**)|(**1**<<**RD\_PIN**)|(**1**<<**DC\_PIN**)|(**1**<<**RST\_PIN**);**

CTRL\_DDR **|=** **(**1**<<**WR\_PIN**)|(**1**<<**RD\_PIN**)|(**1**<<**DC\_PIN**)|(**1**<<**RST\_PIN**);**

RESET\_RST**;** //Zerowanie sprzętowe sterownika SSD1327

\_delay\_ms**(**10**);**

SET\_RST**;**

\_delay\_ms**(**10**);**

writeCommand**(**SET\_COLUMN\_ADDR**);**

writeCommand**(**0x00**);** //Start Column Address

writeCommand**((**OLED\_WIDTH**/**2**)-**1**);** //End Column Address

writeCommand**(**SET\_ROW\_ADDR**);**

writeCommand**(**0x00**);** //Start Row Address

writeCommand**(**OLED\_HEIGHT**-**1**);** //End Row Address

writeCommand**(**SET\_CONTRAST\_CTRL**);**

writeCommand**(**0x9B**);** //Contrast Level

writeCommand**(**SET\_REMAP**);**

writeCommand**(**COLUMN\_ADDR\_REMAP**|**COM\_REMAP**|**COM\_SPLIT\_ODD\_EVEN**);**

writeCommand**(**SET\_DISP\_START\_LINE**);** //Set Display Start Line

writeCommand**(**0x00**);**

writeCommand**(**SET\_DISP\_OFFSET**);**

writeCommand**(**0x00**);**

writeCommand**(**SET\_DISP\_MODE\_NORMAL**);**

writeCommand**(**SET\_MULTIPLEX\_RATIO**);**

writeCommand**(**0x7F**);** //Multiplex = 128

writeCommand**(**SET\_FUNC\_SELECTION\_A**);**

writeCommand**(**ENABLE\_INTERNAL\_VDD**);**

writeCommand**(**SET\_PHASE\_LENGTH**);**

writeCommand**(**0xF1**);** //05

writeCommand**(**SET\_DISP\_CLOCK**);** //Set Display Clock Divide Ratio/Oscillator Frequency

writeCommand**(**0x00**);**

writeCommand**(**SET\_PRECHARGE\_VOLTAGE**);** //Set Prechange Voltage

writeCommand**(**0x07**);** //0.613 x VCC

writeCommand**(**SET\_VCOMH\_VOLTAGE**);** //Set VCOMH Voltage

writeCommand**(**0x07**);** //0.86 x VCC

writeCommand**(**SET\_SEC\_PRECHARGE\_PERIOD**);** //Set Second Pre-charge period

writeCommand**(**0x0F**);**

writeCommand**(**SET\_FUNC\_SELECTION\_B**);**

writeCommand**(**INTERNAL\_VSL**|**ENABLE\_SECOND\_PRECHARGE**);**

writeCommand**(**SET\_GRAY\_SCALE\_TABLE**);**

writeCommand**(**0x01**);** //A[2:0]

writeCommand**(**0x01**);** //B[2:0]

writeCommand**(**0x10**);** //B[6:4]

writeCommand**(**0x01**);** //C[2:0]

writeCommand**(**0x10**);** //C[6:4]

writeCommand**(**0x01**);** //D[2:0]

writeCommand**(**0x10**);** //D[6:4]

writeCommand**(**0x01**);** //E[2:0]

writeCommand**(**0x10**);** //E[6:4]

writeCommand**(**0x01**);** //F[2:0]

writeCommand**(**0x10**);** //F[6:4]

writeCommand**(**0x01**);** //G[2:0]

writeCommand**(**0x10**);** //G[6:4]

writeCommand**(**0x01**);** //H[2:0]

writeCommand**(**0x10**);** //H[6:4]

writeCommand**(**SET\_DISPLAY\_ON**);** //Set Display On

**}**

**Listing 3. Plik nagłówkowy sterownika SSD1327 wyświetlacza WEO128128**

#define DATA\_PORT PORTB

#define DATA\_DDR DDRB

#define CTRL\_PORT PORTD

#define CTRL\_DDR DDRD

#define WR\_PIN PD6

#define RD\_PIN PD7

#define DC\_PIN PD5

#define RST\_PIN PD4

#define RESET\_DC CTRL\_PORT &= ~(1<<DC\_PIN)

#define SET\_DC CTRL\_PORT |= (1<<DC\_PIN)

#define RESET\_WR CTRL\_PORT &= ~(1<<WR\_PIN)

#define SET\_WR CTRL\_PORT |= (1<<WR\_PIN)

#define RESET\_RST CTRL\_PORT &= ~(1<<RST\_PIN)

#define SET\_RST CTRL\_PORT |= (1<<RST\_PIN)

//Rozdzielczość ekranu (pozioma i pionowa)

#define OLED\_WIDTH 128

#define OLED\_HEIGHT 128

//Definicje komend sterujących

#define SET\_COLUMN\_ADDR 0x15

#define SET\_ROW\_ADDR 0x75

#define SET\_CONTRAST\_CTRL 0x81

#define SET\_REMAP 0xA0

#define COLUMN\_ADDR\_REMAP (1<<0)

#define NIBBLE\_REMAP (1<<1)

#define HORIZONTAL\_ADDR\_INCR (0<<2)

#define VERTICAL\_ADDR\_INCR (1<<2)

#define COM\_REMAP (1<<4)

#define COM\_SPLIT\_ODD\_EVEN (1<<6)

#define SET\_DISP\_START\_LINE 0xA1

#define SET\_DISP\_OFFSET 0xA2

#define SET\_DISP\_MODE\_NORMAL 0xA4

#define SET\_DISP\_MODE\_INVERSE 0xA7

#define SET\_MULTIPLEX\_RATIO 0xA8

#define SET\_FUNC\_SELECTION\_A 0xAB

#define SELECT\_EXTERNAL\_VDD (0<<0)

#define ENABLE\_INTERNAL\_VDD (1<<0)

#define SET\_DISPLAY\_OFF 0xAE

#define SET\_DISPLAY\_ON 0xAF

#define SET\_PHASE\_LENGTH 0xB1

#define SET\_DISP\_CLOCK 0xB3

#define SET\_SEC\_PRECHARGE\_PERIOD 0xB6

#define SET\_GRAY\_SCALE\_TABLE 0xB8

#define SET\_PRECHARGE\_VOLTAGE 0xBC

#define SET\_VCOMH\_VOLTAGE 0xBE

#define SET\_FUNC\_SELECTION\_B 0xD5

#define INTERNAL\_VSL (0x60|(0<<0))

#define ENABLE\_EXTERNAL\_VSL (0x60|(1<<0))

#define DISABLE\_SECOND\_PRECHARGE (0x60|(0<<1))

#define ENABLE\_SECOND\_PRECHARGE (0x60|(1<<1))

**Listing 4. Funkcja ustawiająca aktywney obszar ekranu**

//X1 i X2: 0...127 (tylko parzyste), Y1 i Y2: 0...127

void setActiveWindow**(**uint8\_t X1**,** uint8\_t Y1**,** uint8\_t X2**,** uint8\_t Y2**)**

**{**

writeCommand**(**SET\_COLUMN\_ADDR**);** //Set Column Address

writeCommand**(**X1**>>**1**);** //Adres startowy aktywnego obszaru pamięci ekranu dla osi X

writeCommand**(**X2**>>**1**);** //Adres końcowy aktywnego obszaru pamięci ekranu dla osi X

writeCommand**(**SET\_ROW\_ADDR**);** //Set Row address

writeCommand**(**Y1**);** //Adres startowy aktywnego obszaru pamięci ekranu dla osi Y

writeCommand**(**Y2**);** //Adres końcowy aktywnego obszaru pamięci ekranu dla osi Y

**}**

**Listing 5. Funkcje wyświetlające wypełniony i pusty prostokąt**

void drawFilledBox**(**uint8\_t X1**,** uint8\_t Y1**,** uint8\_t X2**,** uint8\_t Y2**,** uint8\_t Color**)**

**{**

uint16\_t Pixels **=** **((**X2**-**X1**+**1**)/**2**)\*(**Y2**-**Y1**+**1**);**

setActiveWindow**(**X1**,** Y1**,** X2**,** Y2**);**

**while(**Pixels**--)** writeData**(**Color**);**

**}**

void drawRectangle**(**uint8\_t X1**,** uint8\_t Y1**,** uint8\_t X2**,** uint8\_t Y2**,** uint8\_t Color**)**

**{**

drawFilledBox**(**X1 **,** Y1 **,** X2 **,** Y1**,** Color**);**

drawFilledBox**(**X1 **,** Y2 **,** X2 **,** Y2**,** Color**);**

drawFilledBox**(**X1 **,** Y1 **,** X1 **,** Y2**,** Color**);**

drawFilledBox**(**X2 **,** Y1 **,** X2 **,** Y2**,** Color**);**

**}**

**Listing 6. Funkcja odpowiedzialna za odczytanie, dekompresję i wyświetlenie obrazka**

void drawPicture**(**uint8\_t X1**,** uint8\_t Y1**,** const uint8\_t **\***Picture**)**

**{**

register uint8\_t byteA**,** byteB**;**

uint16\_t bytesToSend**,** repeats**,** i**=**2**;**

//Obliczamy liczbę bajtów jakie należy wysłać do sterownika SSD1327

bytesToSend **=** **(**pgm\_read\_byte**(&**Picture**[**0**])>>**1**)\***pgm\_read\_byte**(&**Picture**[**1**]);**

//Ustawiamy aktywne okno pamięci obrazu sterownika SSD1327 by uprościć zapis danych

setActiveWindow**(**X1**,** Y1**,** X1**+**pgm\_read\_byte**(&**Picture**[**0**])-**1**,** Y1**+**pgm\_read\_byte**(&**Picture**[**1**])-**1**);**

**do**

**{**

**if(**bytesToSend **==** 1**)** //Dla ostatniego bajta danych

**{**

writeData**(**pgm\_read\_byte**(&**Picture**[**i**]));**

bytesToSend**--;**

**}**

**else**

**{**

byteA **=** pgm\_read\_byte**(&**Picture**[**i**]);** //Odczytujemy bajt "i"

byteB **=** pgm\_read\_byte**(&**Picture**[**i**+**1**]);** //Odczytujemy bajt "i+1"

**if(** byteA **!=** byteB**)** //Sprawdzamy czy kolejne bajty są różne

**{**

writeData**(**byteA**);**

bytesToSend**--;**

i**++;**

**}**

**else**

//Jeśli kolejne bajty są takie same to ustalamy liczbę powtórzeń danego bajta (+2 powtórzenia

//obligatoryjne) wynikające ze sposobu zapisu kompresji typu mRLE (zmodyfikowane RLE)

**{**

repeats **=** pgm\_read\_byte**(&**Picture**[**i**+**2**])+**2**;** //Odczytujemy liczbę powtórzeń danego bajta + 2 bajty obligatoryjne

**while(**repeats**--)** **{**writeData**(**byteA**);** bytesToSend**--;}** //Wysyłamy liczbę powtórzeń bajta

i **+=**3**;** //Przesuwamy indeks o 3 miejsca

**}**

**}**

**}** **while(**bytesToSend**);**

**}**

**Listing 7. Definicja typu danych przechowującego parametry czcionki**

**typedef** struct //Deklaracja struktury przechowującej parametry bieżącej czcionki

**{**

uint8\_t Width**;** //Rzeczywista szerokość znaku (px)

uint8\_t Height**;** //Rzeczywista wysokość znaku (px)

uint8\_t Interspace**;** //Odstęp pomiędzy znakami (px)

uint8\_t BytesPerChar**;** //Liczba bajtów danych tablicy wzorców na 1 znak

uint8\_t FirstCharCode**;** //Kod ASCII pierwszego znaku

const uint8\_t **\***Bitmap**;** //Wskaźnik to tablicy zawierającej wzorce poszczególnych znaków

**}** fontDescription**;**

**Listing 8. Funkcja wybierająca bieżącą czcionkę ekranową**

void setFont**(**const fontDescription **\***Font**)**

**{**

CurrentFont**.**Width **=** pgm\_read\_byte**(&**Font**->**Width**);** //Rzeczywista szerokość czcionki

CurrentFont**.**Height **=** pgm\_read\_byte**(&**Font**->**Height**);** //Rzeczywista wysokość czcionki

CurrentFont**.**Interspace **=** pgm\_read\_byte**(&**Font**->**Interspace**);** //Odstęp pomiędzy znakami

CurrentFont**.**BytesPerChar **=** pgm\_read\_byte**(&**Font**->**BytesPerChar**);** //Liczba bajtów na definicje pojedyn. znaku

CurrentFont**.**FirstCharCode **=** pgm\_read\_byte**(&**Font**->**FirstCharCode**);** //Kod ASCII definicji pierwszego znaku

CurrentFont**.**Bitmap **=** **(**uint8\_t**\*)**pgm\_read\_word**(&**Font**->**Bitmap**);** //Wskaźnik do tablicy wzorców tej czcionki

**}**

**Listing 9. Funkcja odpowiedzialna za rysowanie znaków przy użyciu czcionki ekranowej**

void drawText**(**uint8\_t X1**,** uint8\_t Y1**,** char **\***Text**,** uint8\_t Color**,** uint8\_t Background**)**

**{**

register char Character**;**

register uint16\_t byteIndex**,** offset**;**

register uint8\_t corrWidth**,** bitIndex**,** readByte**;**

//Korygujemy docelową szerokość w celu poprawnego ustawienia okna zapisu, które to zaokrąglamy do pełnych bajtów. Ma to uprościć

//sprawdzenie poszczególnych bitów bajta i tak np. dla szerokości = 6, skorygowana szerokość będzie równa 8, dla 12 równa 16 itd.

corrWidth **=** **((**CurrentFont**.**Width**/**8**)** **+** **(**CurrentFont**.**Width**%**8 **!=**0 **?** 1 **:** 0**))** **\***8**;**

**while** **((**Character **=** **\*(**Text**++)))** //Kolejne znaki napisu wejściowego

**{**

setActiveWindow**(**X1**,** Y1**,** X1**+**corrWidth**-**1**,** Y1**+**CurrentFont**.**Height**-**1**);**

offset **=** **(**Character**-**CurrentFont**.**FirstCharCode**)\***CurrentFont**.**BytesPerChar**;** //Offset położenia wzorca w tablicy wzorców znaków

**for(** byteIndex **=** offset**;** byteIndex **<** **(**offset **+** CurrentFont**.**BytesPerChar**);** byteIndex**++)**

**{**

//Odczytujemy kolejny bajt definicji znaku umieszczony w tablicy Bitmap

readByte **=** pgm\_read\_byte**(**CurrentFont**.**Bitmap**+**byteIndex**);**

//Kolejne pary bitów bajta: 0-1, 2-3, 4-5, 6-7 by utworzyć bajt wynikowy zawierający dane 2 pixeli w 16 odcieniach

**for(** bitIndex**=**0**;** bitIndex**<**6**;** bitIndex**+=**2 **)**

**{**

**if(**readByte **&** **(**1**<<**bitIndex**))** corrWidth**=** Color **<<**4**;** **else** corrWidth **=** Background **<<**4**;** //corrWidth użyte jako temp

**if(**readByte **&** **(**1**<<(**bitIndex**+**1**)))** corrWidth **|=** Color**;** **else** corrWidth **|=** Background**;**

writeData**(**corrWidth **);**

**}**

**}**

X1**+=(**CurrentFont**.**Width**+**CurrentFont**.**Interspace**);** //Przesuwamy się o rzeczywistą szerokość znaku plus odstęp pomiędzy znakami

**}**

**}**